11-02-00   A

UTILITY PATENT APPLICATION TRANSMITTAL (Large Entity) *(Only for new nonprovisional applications under 37 CFR 1.53(b))*	Docket No. 13734 (YOR920000357US1)
	Total Pages in this Submission

TO THE ASSISTANT COMMISSIONER FOR PATENTS
Box Patent Application
Washington, D.C. 20231

Transmitted herewith for filing under 35 U.S.C. 111(a) and 37 C.F.R. 1.53(b) is a new utility patent application for an invention entitled:

SYSTEM AND METHOD FOR CHARACTERIZING PROGRAM BEHAVIOR BY SAMPLING AT SELECTED PROGRAM POINTS

and invented by:

Matthew R. Arnold   Peter F. Sweeney
Stephen J. Fink
David P. Grove
Michael J. Hind

If a CONTINUATION APPLICATION, *check appropriate box and supply the requisite information:*

❑ **Continuation** ❑ **Divisional** ❑ **Continuation-in-part (CIP)** of prior application No.: _____

Which is a:

❑ **Continuation** ❑ **Divisional** ❑ **Continuation-in-part (CIP)** of prior application No.: _____

Which is a:

❑ **Continuation** ❑ **Divisional** ❑ **Continuation-in-part (CIP)** of prior application No.: _____

Enclosed are:

Application Elements

1. ☒ Filing fee as calculated and transmitted as described below

2. ☒ Specification having _____ 22 _____ pages and including the following:

   a. ☒ Descriptive Title of the Invention

   b. ❑ Cross References to Related Applications *(if applicable)*

   c. ❑ Statement Regarding Federally-sponsored Research/Development *(if applicable)*

   d. ❑ Reference to Microfiche Appendix *(if applicable)*

   e. ☒ Background of the Invention

   f. ☒ Brief Summary of the Invention

   g. ❑ Brief Description of the Drawings *(if drawings filed)*

   h. ☒ Detailed Description

   i. ☒ Claim(s) as Classified Below

   j. ☒ Abstract of the Disclosure

UTILITY PATENT APPLICATION TRANSMITTAL **(Large Entity)** *(Only for new nonprovisional applications under 37 CFR 1.53(b))*	Docket No. 13734 (YOR920000357US1)
	Total Pages in this Submission

Application Elements (Continued)

3. ☐ Drawing(s) *(when necessary as prescribed by 35 USC 113)*

    a. ☐ Formal   Number of Sheets   _____

    b. ☐ Informal   Number of Sheets   _____

4. ☒ Oath or Declaration

    a. ☒ Newly executed *(original or copy)*   ☐ Unexecuted

    b. ☐ Copy from a prior application (37 CFR 1.63(d)) *(for continuation/divisional application only)*

    c. ☒ With Power of Attorney   ☐ Without Power of Attorney

    d. ☐ *DELETION OF INVENTOR(S)*
       Signed statement attached deleting inventor(s) named in the prior application,
       see 37 C.F.R. 1.63(d)(2) and 1.33(b).

5. ☐ Incorporation By Reference *(usable if Box 4b is checked)*
    The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.

6. ☐ Computer Program in Microfiche *(Appendix)*

7. ☐ Nucleotide and/or Amino Acid Sequence Submission *(if applicable, all must be included)*

    a. ☐ Paper Copy

    b. ☐ Computer Readable Copy *(identical to computer copy)*

    c. ☐ Statement Verifying Identical Paper and Computer Readable Copy

Accompanying Application Parts

8. ☒ Assignment Papers *(cover sheet & document(s))*

9. ☐ 37 CFR 3.73(B) Statement *(when there is an assignee)*

10. ☐ English Translation Document *(if applicable)*

11. ☒ Information Disclosure Statement/PTO-1449   ☒ Copies of IDS Citations

12. ☐ Preliminary Amendment

13. ☒ Acknowledgment postcard

14. ☒ Certificate of Mailing

    ☐ First Class   ☒ Express Mail *(Specify Label No.):* EL658969695US

UTILITY PATENT APPLICATION TRANSMITTAL
(Large Entity)
(Only for new nonprovisional applications under 37 CFR 1.53(b))

Docket No.
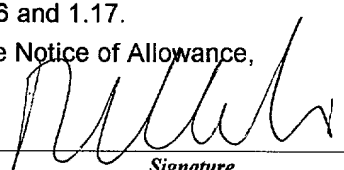13734 (YOR920000357US1)

Total Pages in this Submission

Accompanying Application Parts (Continued)

15. ❑ Certified Copy of Priority Document(s) *(if foreign priority is claimed)*

16. ❑ Additional Enclosures *(please identify below):*

Associate Power of Attorney and Request for Change of Mailing Address

Request That Application Not Be Published Pursuant To 35 U.S.C. 122(b)(2)

17. ❑ Pursuant to 35 U.S.C. 122(b)(2), Applicant hereby requests that this patent application not be published pursuant to 35 U.S.C. 122(b)(1). Applicant hereby certifies that the invention disclosed in this application has not and will not be the subject of an application filed in another country, or under a multilateral international agreement, that requires publication of applications 18 months after filing of the application.

Warning

An applicant who makes a request not to publish, but who subsequently files in a foreign country or under a multilateral international agreement specified in 35 U.S.C. 122(b)(2)(B)(i), must notify the Director of such filing not later than 45 days after the date of the filing of such foreign or international application. A failure of the applicant to provide such notice within the prescribed period shall result in the application being regarded as abandoned, unless it is shown to the satisfaction of the Director that the delay in submitting the notice was unintentional.

P01ULRG/REV05

UTILITY PATENT APPLICATION TRANSMITTAL
(Large Entity)
(Only for new nonprovisional applications under 37 CFR 1.53(b))

Docket No. 13734 (YOR920000357US1)	
Total Pages in this Submission	

Fee Calculation and Transmittal

CLAIMS AS FILED

For	#Filed	#Allowed	#Extra	Rate	Fee
Total Claims	36	- 20 =	16	x $18.00	$288.00
Indep. Claims	3	- 3 =	0	x $80.00	$0.00
Multiple Dependent Claims (check if applicable) ☐					$0.00
				BASIC FEE	$710.00
OTHER FEE *(specify purpose)*					$0.00
				TOTAL FILING FEE	$998.00

☐ A check in the amount of _____ to cover the filing fee is enclosed.

☒ The Commissioner is hereby authorized to charge and credit Deposit Account No. **50-0510/IBM** as described below. A duplicate copy of this sheet is enclosed.

    ☒ Charge the amount of **$998.00** as filing fee.

    ☒ Credit any overpayment.

    ☒ Charge any additional filing fees required under 37 C.F.R. 1.16 and 1.17.

    ☐ Charge the issue fee set in 37 C.F.R. 1.18 at the mailing of the Notice of Allowance, pursuant to 37 C.F.R. 1.311(b).

_____
Signature

Richard L. Catania
Registration No. 32,608
SCULLY, SCOTT, MURPHY & PRESSER
400 Garden City Plaza
Garden City, NY 11530
(516) 742-4343

Dated: **November 1, 2000**

cc:

P01ULRG/REV05

CERTIFICATE OF MAILING BY "EXPRESS MAIL" (37 CFR 1.10)	Docket No.
Applicant(s): Matthew r. Arnold, et al.	13734 (YOR920000357US1)

Serial No. unassigned	Filing Date herewith	Examiner	Group Art Unit unassigned

Invention: **SYSTEM AND METHOD FOR CHARACTERIZING PROGRAM BEHAVIOR BY SAMPLING AT SELECTED PROGRAM POINTS**

I hereby certify that the following correspondence:

New Patent Application

(Identify type of correspondence)

is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under

37 CFR 1.10 in an envelope addressed to: The Assistant Commissioner for Patents, Washington, D.C. 20231

November 1, 2000
(Date)

Mishelle Mustafa
(Typed or Printed Name of Person Mailing Correspondence)

(Signature of Person Mailing Correspondence)

EL658969695US
("Express Mail" Mailing Label Number)

Note: Each paper must have its own certificate of mailing.

P06A/REV02

SYSTEM AND METHOD FOR CHARACTERIZING PROGRAM BEHAVIOR BY SAMPLING AT SELECTED PROGRAM POINTS

BACKGROUND OF THE INVENTION

Field of the Invention

5

This invention relates generally to computer program execution systems, e.g., optimizing compilers, and more specifically, to a system and method for sampling executing programs at selected

10 program yield points to enable the characterization of runtime program behavior.

Discussion of the Prior Art

15 Many modern programming language runtime environments and tools can benefit from runtime feedback from a program. For example, Java virtual machines may use runtime feedback to guide optimization of the running program. As another example, program understanding tools may gather runtime information and report

20 summaries to the user.

Since running programs may potentially generate vast quantities of runtime data, many systems use *statistical sampling* to reduce the volume of information. With this well-known technique, the

25 system collects only a subset, or *sample*, of the total relevant runtime information, and infers properties of the program by extrapolating from the sample.

Although sampling is a well-known principle, a system that

30 implements sampling faces potentially difficult engineering

1

tradeoffs. The system must implement a sampling mechanism that gathers an interesting subset of the data, while minimizing runtime overhead. For some applications, the system must implement a mechanism that collects multiple independent samples.

5 The system must extrapolate from the sampled data to recover the information it desires, a non-trivial task in some cases. Also, for some applications, the system must rely on compiler support to provide information about the program, and integrate this support with the runtime system.

10

Many compilers and programming tools collect runtime information to characterize the behavior of a running program. In order to collect runtime information, the system must periodically interrupt the running program and record information regarding

15 the current state of execution.

There are two previous approaches to interrupting programs to collect runtime information. In the first approach, the system interrupts the program at any arbitrary instruction. For example,

20 the DCPI profiling tool described in the reference to J.M. Andersen, L.M. Berc, J.Dean, et al. entitled "Continuous profiling: Where have all the cycles gone?", Technical Note 1997-016a, Digital Systems Research Center, www.research.digital.com/SRC, Sept. 1997, interrupts the program

25 after executing a fixed number of instructions. An advantage of this approach is that the mechanism works for any binary program, and requires no participation by the compiler.

In the second approach, the system identifies distinguished

30 program points at which it collects information. For example,

2

the IBM family MMI Development Kits for Java supports invocation counters at method entries. With this approach, the compiler or interpreter conspires with the profiling system to interrupt the program at particular points. The advantage of this approach is

5 that the compiler or interpreter can record detailed information specific to these distinguished program points. So, for example, the MMI systems record the identity of the interrupted method at each invocation counter point.

10 It would be highly desirable to provide improved methods and mechanisms for collecting executing program runtime information. In particular it would be highly desirable to provide improved methods and mechanisms for collecting executing program runtime information at a subset of distinguished program points in a

15 manner so as to reduce runtime overhead.

Summary of the Invention

It is an object of the present invention to provide improved

20 methods and mechanisms for collecting executing program runtime information.

It is a further object of the present invention to provide improved methods and mechanisms for collecting executing program

25 runtime information at a subset of distinguished program points in a manner so as to reduce runtime overhead.

According to the invention, there is provided a system and method for collecting information at a subset of distinguished program

30 points, and particularly, a mechanism to collect a statistical

3

sample of the information that would be collected at all
identified program points. One potential advantage is that by
using statistical sampling, the invention will reduce runtime
overhead compared to exhaustive sampling at distinguished program
5   points.

Advantageously, such a system and method is general enough to be
applied to compiler and interpreter run-time processing
environments. That is, similar mechanisms also apply when running
10 interpreted code, as will be apparent to those skilled in the
art.

Detailed Description of the Preferred Embodiments

15 For exemplary purposes, the present invention is described for
operation in a particular JVM targeting server applications that
implement a "compile-only" strategy by compiling all methods to
native code before they execute, such as described in the
references "Jalapeno Virtual Machine", IBM Systems Journal,
20 39(1), 2000 by B.Alpern, C. R. Attanasio, et al and "Implementing
Jalapeno in Java", ACM Conference on Object-Oriented Programming
Systems, Languages, and Applications, 1999, both of which are
incorporated by reference as if fully set forth herein. However,
it is understood that the principles of program characterization
25 as described herein may be applicable for any run-time
environment, e.g., JVM, interpreters, Just-in-Time compilers,
etc.

In the JVM, Java threads are multiplexed onto operating system
30 threads. The underlying operating system in turn maps pthreads

4

to physical processors (CPUs). At any given moment in time, each virtual processor may have any number of Java threads assigned to it for execution. The system supports thread scheduling with a quasi-preemptive mechanism. Further, each compiler generates

5 *yield points*, which are program points where the running thread checks a dedicated bit in a machine control register to determine if it should yield the virtual processor. Currently, the compilers insert these yield points in method prologues and on loop back edges. As known, algorithms exist that optimize

10 placement of yield points to reduce the dynamic number of yield points executed while still supporting effective quasi-preemptive thread scheduling. Using a timer-interrupt mechanism, an interrupt handler periodically sets a bit on all virtual processors. When a running thread next reaches a yield point, a

15 check of the bit will result in a call to the scheduler.

It is assumed that information is being collected from a compiled binary program, however, it is understood that mechanisms in accordance with the principles of the invention may be employed

20 for running interpretive code.

According to the principles of the invention, a trigger is defined as a bit of program state that specifies whether an action should be taken. The run-time system or instrumented code

25 may set a trigger to signal that an action should be taken. Further, a yield point is defined as a special sequence of instructions that performs the following actions when it is executed: 1) it checks the trigger bit; 2) if the trigger bit is set, the yield point is taken and some action is performed; and

30 3) if the trigger bit is not set, the yield point is not taken,

5

no action is performed, and the next instruction after the yield point is executed.

At a high-level, the invention's method comprises the following
5   steps: The compiler inserts yield points at distinguished program points. At runtime, the system periodically sets off a trigger when it decides to take a sample of current program behavior. When the running program next encounters a yield point, it observes that the trigger has been set and takes some action.
10

The actions performed at yield points occur at a subset of the executions of yield points. Depending on the type of information and sampling technique desired, the system implementer may choose from a variety of policies regarding where to insert yield
15   points, when to set triggers, and what action a yield point should take.

With regard to the placement of yield points, although yield points may be placed at an arbitrary subset of program points,
20   the preferred embodiment places yield points in all method prologues and in all loop headers (a back edge). As, in some circumstances, identifying loop headers may incur unacceptable levels of overhead, an alternative placement of yield points in all method prologues and at the targets of all backwards intra-
25   procedural branches may be used instead. In either case, the system distinguishes between prologue yield points and loop yield points and may take different sampling actions when a yield point is taken in a method prologue rather than when a yield point is taken in a loop.
30

6

Abstractly, a prologue yield point performs the following system operations represented by the following pseudocode:

```
if (shouldTakePrologueYieldPoint) then
    takePrologueSample()
end
```

Similarly, a loop yield point performs the following system operations:

```
if (shouldTakeLoopYieldPoint) then
    takeLoopSample()
end
```

A preferred embodiment implements a timer based approach. Preferably, associated with **shouldTakePrologueYieldPoint** and **shouldTakeLoopYieldPoint** is the reserved bit Atrigger bit@ which is initially set to 0. Using standard operating system signal mechanisms, an interrupt is arranged to occur at periodic time intervals. An interrupt handler is coded to catch the timer interrupt. When the handler catches the interrupt, it sets the trigger bit to be 1. Yield points check the value of the trigger bit, and when it is 1 the yield point is taken, a sample is collected, and the trigger bit is reset to 0. In this implementation, the pseudo code for prologue yield points is as follows:

```
if (triggerBit == 1) then
    takePrologueSample()
    triggerBit = 0
end
```

7

Similarly, the pseudo code for loop yield points is as follows:

```
if (triggerBit == 1) then
    takeLoopSample()
    triggerBit = 0;
end
```

In some architectures, an efficient implementation may be to dedicate a bit in one of the CPU's condition registers to hold the trigger bit.

An alternative to the timer-based approach is use of a decrementing counter to arrange that a fixed percentage of all executed yield points are taken. For example, an implementation of the counter-based approach is given by the following pseudo-code for prologue yield points:

```
if (yieldPointCounter == 0) then
    takePrologueSample()
    yieldPointCounter = numYieldPointsToSkip;
else
    yieldPointCounter = yieldPointCounter - 1;
end
```

Similarly, the pseudo-code for loop yield points for this approach is:

```
if (yieldPointCounter == 0) then
    takeLoopSample()
    yieldPointCounter = numYieldPointsToSkip;
else
    yieldPointCounter = yieldPointCounter - 1;
end
```

8

As will be appreciated by those skilled in the art, a counter-based yield point taking mechanism may be efficiently implemented on hardware architectures such as the PowerPC that include a count register and a decrement and conditional branch

5 on count instruction.

A third approach blends the first two implementations by using a combined counter and timer based yield points in method prologues with a timer only yield point in loops. This may be desirable to

10 support profile-directed inlining in the manner as described in commonly-owned, co-pending U.S. Patent application No. _____ (YOR920000358, D#13735), the contents and disclosure of which are incorporated by reference herein. An implementation of this approach is given by the following pseudo-code for prologue yield

15 points:
```
if (triggerBit == 1 || yieldPointCounter == 0) then
    takePrologueSample()
    if (triggerBit)
        triggerBit = 0;
20  end
    if (yieldPointCounter == 0)
        yieldPointCounter = numYieldPointsToSkip;
    end
else
25  yieldPointCounter = yieldPointCounter - 1;
end
```

For loop yield points a pseudocode implementation is as follows:

```
30 if (triggerBit == 1) then
    takeLoopSample()
    triggerBit = 0;
end
```

9

Again, those skilled artisans will appreciate that the above prologue yield point may be efficiently implemented on architectures with a count register and associated machine instructions.

5

In accordance with the invention, it is understood that a wide variety of sampling information may be collected when a yield point is taken. That is, a low-level mechanism exists that is available to map from a taken yield point to a method. Typical

10 mechanisms include (1) inspecting the hardware state to determine the instruction address at which the yield point was taken and mapping that address to a method; and (2) inspecting the program's runtime stack to identify the method in which the yield point was taken, possibly by inspecting the return addresses

15 stored on the runtime stack. These low-level sampling mechanisms identify and track executing methods with the frequency of executed methods being recorded for characterizing program behavior. In a similar manner, further information such as thecall-context, frequency of executing basic blocks and program

20 variable values may be recorded for characterizing program behavior.

Implementations of **takePrologueSample** and **takeLoopSample** are now provided. One implementation of **takePrologueSample** and

25 **takeLoopSample** comprises determining which method was executing when the yield point was taken and incrementing a counter associated with that method. If the yield point was taken in a loop, then the sample should be attributed to the method containing the loop. If the yield point was taken in a prologue,

30 then the sample may be attributed to the calling method, the

10

called method, or to both the calling and called method. A
preferred embodiment is to attribute 50% of a sample to each of
the caller and callee methods.

5   In addition to incrementing a method counter, more complex
samples may be taken to aid method inlining. For example, the
techniques described in commonly-owned, co-pending U.S. Patent
Application No. _____ (YOR9200000357, D#13732) entitled METHOD
FOR CHARACTERIZING PROGRAM EXECUTION BY PERIODIC CALL-STACK
10 INSPECTION, the contents and disclosure of which is incorporated
by reference as if fully set forth herein, are potential
embodiments for **takePrologueSample**, and may be used for
ascertaining call-context of executing program methods.

15 Depending on the type of information and sampling technique
desired, the system implementer can choose from a variety of
policies regarding where to insert yield points, when to set
triggers, and what action a yield point should take. A concrete
example of this procedure to gather a statistical sample of
20 method invocation behavior is now described.

In an example embodiment, the system collects a statistical
sample of all method invocations. To use the invention, three
considerations are addressed: 1) Where to insert yield points?;
25 2) When to set the trigger?; and 3) What action should be
performed when a yield point is taken?

For this example, yield points are inserted in every method
prologue with the trigger bit set off periodically based on an
30 external timer. Further, for this example, the action will tally

11

the number of times it is invoked from each method prologue by
calling a Listener routine which records data periodically or
when a sampling condition is determined.

5 Having specified the policies, a preferred embodiment of the
mechanisms employed to implement these policies is now given.

The trigger is implemented by reserving a single bit in the
computer system's memory. Initially, this bit is set to 0.
10 Using standard operating system signal mechanisms, an interrupt
is arranged to occur at periodic time intervals with an interrupt
handler coded to catch the timer interrupt. When the handler
catches the interrupt, it sets the trigger bit to be 1. It is
assumed that the system assigns each method in the program a
15 unique integer identifier, called the method id. The yield point
sequence of instructions, in pseudo-code, are as follows:

```
if (trigger bit == 1) then
    call Listener(current method id)
20  end
```

Finally, we describe the implementation of the Listener routine.
It keeps a table of integers, indexed by method id. Name this
table the MethodCount table. The Listener routine simply
25 increments the MethodCount table for the method i.d. that it is
passed, and clears the trigger bit, as follows:

```
subroutine Listener(method id)
    increment MethodCount table entry for method id
30  trigger bit = 0
```

12

This completes the preferred embodiment for this example.
Naturally, the system will later process the statistical
information collected as needed, as will be obvious to those
skilled in the art.

5

While the invention has been particularly shown and described
with respect to illustrative and preformed embodiments thereof,
it will be understood by those skilled in the art that the
foregoing and other changes in form and details may be made
10 therein without departing from the spirit and scope of the
invention which should be limited only by the scope of the
appended claims.

13

YOR920000357US1

CLAIMS:

Having thus described our invention, what we claim as new, and
desire to secure by Letters Patent is:

1   1. A method for characterizing runtime behavior of a computer
2   program executing in an execution environment, said method
3   comprising:
4   a) identifying one or more instances of yield points in
5   a program to be executed, each said yield point indicating a
6   potential sampling operation during execution of said program;
7   b) during program execution, in response to an
8   identified yield point instance, ascertaining a state of said
9   execution environment for indicating whether a sampling operation
10 is to be performed; and,
11   c) when state of said execution environment indicates a
12 sampling operation, recording relevant information for
13 characterizing behavior of said execution environment.

1   2. The method as claimed in Claim 1, wherein said sampling
2   operation includes identifying a method currently executing in
3   said program, said method including tracking frequencies of
4   methods executed in said program for characterizing said program
5   behavior.

1   3. The method as claimed in Claim 2, wherein said sampling
2   operation includes identifying a calling context associated with
3   methods called by said program, said method including tracking
4   calling context frequency for characterizing said program
5   behavior.

14

1   4. The method as claimed in Claim 1, wherein said sampling

2   operation includes identifying current program variable values,

3   said program variable values being tracked for characterizing

4   said program behavior.

1   5. The method as claimed in Claim 1, wherein said sampling

2   operation includes identifying basic blocks executed in said

3   program, said method including tracking a frequency of basic

4   blocks for characterizing said program behavior.

1   6. The method as claimed in Claim 1, wherein when said state of

2   said execution environment does not indicate a sampling

3   operation, the step of executing a next instruction in said

4   executing program after said identified yield point.

1   7. The method as claimed in Claim 1, wherein said step b) of

2   ascertaining a state of said execution environment includes

3   checking status of a trigger bit set by said execution

4   environment to indicate performance of said sampling operation.

1   8.   The method as claimed in Claim 1, wherein said trigger bit

2   status is set periodically by said executing environment.

1   9.   The method as claimed in Claim 8, further including the steps

2   of:

3   invoking a runtime system interrupt at periodic time

4   intervals; and,

5   implementing an interrupt handler mechanism for

6   catching said interrupt and setting said trigger bit.

15

YOR920000357US1

1   10.   The method as claimed in Claim 2, wherein said step of

2   identifying a currently executing method comprises determining an

3   instruction address at which the yield point was taken and

4   mapping that address to a called method.

1   11.   The method as claimed in Claim 3, wherein said step of

2   identifying a calling context associated with methods comprises

3   inspecting a call-stack runtime data structure for tracking

4   methods currently active in said executing program.

1   12.   The method as claimed in Claim 1, further including the step

2   of implementing a compiler device for inserting one or more yield

3   points in said program.

1   13.   The method as claimed in Claim 1, further including the step

2   of implementing an interpreter device for ensuring execution of

3   said yield points in said program.

1   14.   The method as claimed in Claim 1, wherein said yield points

2   are inserted in one or more program locations including: a method

3   prologue and a loop back edge.

1   15. A method for characterizing runtime behavior of a computer

2   program executing in an execution environment, said method

3   comprising:

4   a) identifying one or more instances of yield points

5   inserted in a executing program, each said yield point indicating

6   a potential sampling operation during execution of said program;

7   b) counting a number of identified yield points;

8   c) comparing said number against a predetermined

9   threshold; and,

10   d) in response to meeting said threshold, performing a
11 sampling operation of said executing program, and, recording
12 relevant information for characterizing behavior of said
13 execution environment in response to said sampling.

1 16. The method as claimed in Claim 15, wherein said sampling
2 operation includes identifying a method currently executing in
3 said program, said method including tracking frequencies of
4 methods executed in said program for characterizing said program
5 behavior.

1 17. The method as claimed in Claim 16, wherein said sampling
2 operation includes identifying a calling context associated with
3 methods called by said program, said method including tracking
4 calling context frequency for characterizing said program
5 behavior.

1 18. The method as claimed in Claim 15, wherein said sampling
2 operation includes identifying current program variable values,
3 said program variable values being tracked for characterizing
4 said program behavior.

1 19. The method as claimed in Claim 15, wherein said sampling
2 operation includes identifying basic blocks executed in said
3 program, said method including tracking a frequency of basic
4 blocks for characterizing said program behavior.

1 20. The method as claimed in Claim 15, wherein said step c)
2 includes the steps of:
3   initializing a counter to said predetermined threshold;
4 and, for each identified yield point instance,

17

5   decrementing said counter until said counter is zero,
6 whereby said sampling operation is arranged such that a fixed
7 percentage of all executed yield points are taken.

1 21. The method as claimed in Claim 16, wherein said step of
2 identifying a currently executed method comprises determining an
3 instruction address at which the yield point was taken and
4 mapping that address to a called method.

1 22. The method as claimed in Claim 17, wherein said step of
2 identifying a calling context associated with methods comprises
3 inspecting a call-stack runtime data structure for tracking
4 methods currently active in said executing program.

1 23. The method as claimed in Claim 15, further including the
2 step of implementing a compiler device for inserting one or more
3 yield points in said program, said yield points being in one or
4 more program locations including: a method prologue and a loop
5 back edge.

1 24. The method as claimed in Claim 15, further including the
2 step of implementing an interpreter device for ensuring execution
3 of said yield points in said program.

1 25. A system for characterizing runtime behavior of a computer
2 program executing in an execution environment, said system
3 comprising:
4   a) mechanism for identifying instances of yield points
5 inserted in an executing program;

18

6   b) control device for determining a condition for

7 performing a sampling operation of said executing program at an

8 identified yield point instance; and,

9   c) sampling device for performing said sampling

10 operation of said executing program upon satisfaction of said

11 condition, and recording relevant information for characterizing

12 behavior of said execution environment in response to said

13 sampling.

1 26. The system as claimed in Claim 25, wherein said sampling

2 device includes mechanism for identifying a method currently

3 executing in said program; said sampling device comprising

4 mechanism for tracking frequencies of methods executed in said

5 program for characterizing said program behavior.

1 27. The system as claimed in Claim 26, wherein said sampling

2 device includes mechanism for identifying a calling context

3 associated with methods called by said program, said tracking

4 mechanism further tracking calling context frequency for

5 characterizing said program behavior.

1 28. The system as claimed in Claim 25, wherein said sampling

2 operation includes mechanism for identifying current program

3 variable values, said tracking mechanism further tracking said

4 program variable values for characterizing said program behavior.

1 29. The system as claimed in Claim 25, wherein said sampling

2 device includes mechanism for identifying basic blocks executed

3 in said program, said tracking mechanism further tracking a

4 frequency of basic blocks for characterizing said program

5 behavior.

1   30. The system as claimed in Claim 25, further including:
2   a system location for storing a trigger bit; and,
3   a runtime system for said executing environment, said
4   runtime system setting said trigger bit to indicate performance
5   of said sampling operation; wherein, said control device
6   ascertains a state of said system bit for determining said
7   sampling condition.

1   31. The system as claimed in Claim 30, wherein said runtime
2   system includes:
3   interrupt mechanism for generating timer interrupt
4   signal; and,
5   interrupt handler mechanism for catching said interrupt
6   and setting said trigger bit.

1   32. The system as claimed in Claim 26, wherein said mechanism
2   for identifying a currently executed method comprises includes
3   determining an instruction address at which the yield point was
4   taken and mapping that address to a called method.

1   33. The system as claimed in Claim 27, wherein said mechanism
2   for identifying a calling context associated with methods
3   comprises inspecting a call-stack runtime data structure for
4   tracking methods currently active in said executing program.

1   34. The system as claimed in Claim 25, wherein said control
2   device comprises:
3   counter device for counting a number of identified
4   yield points; and,

20

5   device for comparing said number against a
6 predetermined threshold value, wherein, in response to meeting of
7 said threshold, said control device initiating performing of said
8 sampling operation.

1 35. The system as claimed in Claim 25, further including a
2 compiler device for inserting one or more yield points in said
3 program.

1 36. The system as claimed in Claim 25, further including an
2 interpreter device for ensuring execution of said yield points in
3 said program.

21

SYSTEM AND METHOD FOR CHARACTERIZING PROGRAM
BEHAVIOR BY SAMPLING AT SELECTED PROGRAM POINTS

5

ABSTRACT OF THE DISCLOSURE

A system and method for characterizing runtime behavior of a
computer program executing in an execution environment, the
10   method comprising: identifying one or more instances of yield
points in a program to be executed, each yield point indicating a
potential sampling operation during program execution; during
program execution, in response to an identified yield point
instance, ascertaining a state of the execution environment for
15   indicating whether a sampling operation is to be performed; and,
when the state of the execution environment indicates a sampling
operation, recording relevant information for characterizing
behavior of the execution environment. Relevant information for
characterizing program behavior includes frequencies of methods
20   executed in the program, and calling context associated with
methods called by the program. Different mechanisms are provided
for determining the sampling condition including the setting of a
trigger bit by a runtime system, or, determining a sampling
operations based on a fixed percentage of all executed yield
25   points taken.

22

YOR920000357US1

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION
As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled: SYSTEM AND METHOD FOR CHARACTERIZING PROGRAM BEHAVIOR BY SAMPLING AT SELECTED PROGRAM POINTS

the specification of which (check one)

_x_ is attached hereto.

____ was filed on _____ as United States Application Number

     or PCT International Application Number

     and was amended on _____ (if applicable)

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, §119(a)-(d) or §365(b) of any foreign application(s) for patent or inventor's certificate, or §365(a) of any PCT International application which designated at least one country other than the United States, listed below and have also identified below, by checking the box, any foreign application for patent or inventor's certificate, or PCT International application, having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)   Priority Claimed

(Number)	(Country)	(Day/Month/Year Filed)	___ Yes ___ No
(Number)	(Country)	(Day/Month/Year Filed)	___ Yes ___ No
(Number)	(Country)	(Day/Month/Year Filed)	___ Yes ___ No

I hereby claim the benefit under 35 U.S.C. §119(e) of any United States provisional application(s) listed below.

(Application Number)   (Filing Date)

(Application Number)   (Filing Date)

I hereby claim the benefit under 35 U.S.C. §120 of any United States Application(s), or §365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States, or PCT International application in the manner provided by the first paragraph of 35 U.S.C. §112, I acknowledge the duty to disclose information material to the patentability of this application as defined in 37 CFR §1.56 which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Serial No.)   (Filing Date)   (Status) (patented, pending, abandoned)

(Application Serial No.)   (Filing Date)   (Status) (patented, pending, abandoned)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that willful false statements may jeopardize the validity of the application or any patent issued thereon.

POWER OF ATTORNEY: As a named inventor I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and transact all business in the Patent and Trademark Office connected therewith (list name and registration number).

Manny W. Schecter (Reg. 31,722), Lauren C. Bruzzone (Reg. No. 35,802), Christopher A. Hughes (Reg. 26,914), Edward A. Pennington (Reg. 32,588), John E. Hoel (Reg. 26,279), Joseph C. Redmond, Jr. (Reg. 18,753), Douglas W. Cameron (Reg. No. 31,596), Wayne L. Ellenbogen (Reg. No. 43,602), Stephen C. Kaufman (Reg. No. 29,551), Daniel P. Morris (Reg. No. 32,053), Louis J. Percello (Reg. No. 33,206), David M. Shofi (Reg. No. 39,835), Robert M. Trepp (Reg. No. 25,933), Paul J. Otterstedt (Reg. No. 37,411) and Louis P. Herzberg (Reg. No. 41,500) and Marian Underweiser (Reg. No. 46,134).

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

Send Correspondence to: Richard L. Catania, Scully, Scott, Murphy & Presser

400 Garden City Plaza, Garden City, New York 11530

Direct Telephone Calls to: (name and telephone number) Richard L. Catania, (516) 742-4343

Matthew R. Arnold
Full name of sole or first inventor

Inventor's Signature _____ Date 10/6/2000

389 Teaneck Road, Ridgefield Park, NJ 07660
Residence

USA
Citizenship

Same as residence
Post Office Address

Stephen J. Fink
Full name of second joint inventor, if any

Inventor's signature _____ Date 10/6/2000

2693 Cecile Drive, Yorktown Heights, NY 10598
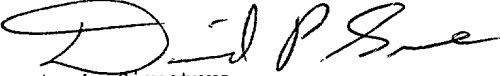Residence

USA
Citizenship

Same as residence
Post Office Address

David P. Grove
Full name of third joint inventor, if any

Inventor's Signature   Date 10/11/2000

97 Tackora Trail, Ridgefield, CT 06877
Residence

USA
Citizenship

Same as residence
Post Office Address

Michael J. Hind
Full name of fourth joint inventor, if any

Inventor's signature   Date 10/6/2000

11 East Hill Road, Cortlandt Manor, NY 10567
Residence

USA
Citizenship

Same as residence
Post Office Address

Peter F. Sweeney
Full name of fifth joint inventor, if any

Inventor's Signature   Date 10/6/2000

30 South Cole Avenue, Spring Valley, NY 10977
Residence

USA
Citizenship

Same as residence
Post Office Address

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants: Matthew R. Arnold et al.   **Docket:** 13734
(YOR920000357US1)

Serial No.: Unassigned   **Dated:**

Filed: Herewith

For: SYSTEM AND METHOD FOR CHARACTERIZING PROGRAM
BEHAVIOR BY SAMPLING AT SELECTED PROGRAM POINTS

Assistant Commissioner for Patents
Washington, DC 20231

ASSOCIATE POWER OF ATTORNEY AND
REQUEST FOR CHANGE OF MAILING ADDRESS

Sir:

Applicants, by their attorneys of record, hereby
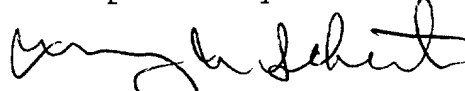grant an Associate Power of Attorney to:

RICHARD L. CATANIA, Reg. No. 32,608; FRANK S. DIGIGLIO, Reg.
31,346; KENNETH L. KING, Reg. No. 24,223; STEPHEN D. MURPHY,
Reg. No. 22,002; LEOPOLD PRESSER, Reg. No. 19,827; and JOHN S.
SENSNY, Reg. No. 28,757

with full power of substitution to prosecute this application
and transact all business in the United States Patent and
Trademark Office in connection therewith.

Applicants further request that all future
correspondence in connection with this application be directed
and addressed to:

RICHARD L. CATANIA, ESQ.
SCULLY, SCOTT, MURPHY AND PRESSER
400 Garden City Plaza
Garden City, New York 11530
Direct all telephone calls to: (516) 742-4343.

Respectfully submitted,

Manny W. Schecter
Registration No.: 31,722
Telephone No.: (914)945-3252

IBM Corporation
T.J. Watson Research Center
Route 134/Kitchawan Road
P.O. Box 218
Yorktown Heights, NY 10598
SF/vjs

G:\work\105\misc\13734.apo